A Review of Strong Gravitational Lensing Tools for SGL Image Modeling

Mason Tea, Summer 2019

Wesleyan University, Department of Astronomy

Introduction

Goals

This paper was compiled in order to gather and assess software tools currently available for the modeling of strong gravitational lens systems. The assessment of these tools is, of course, based on the requirements of the software we hope to produce for use with the SGL. In general terms, these tools include:

- A solar Einstein ring modeling program which takes pre-generated images as input (EPIC DSCOVR data, for example) and returns physically accurate, strongly-lensed deformations of these images, as the SGL might see if the original object were in syzygy with the Sun,
- A lensed-image deconvolution program which takes the aforementioned deformed images and re-generates the original images based on the information contained in the deformed ones.

The current goal is to use the data from the first program to train the second, which in theory would contain a neural net for this purpose, but we are open to other formulations in this brainstorming stage. This is why we seek to survey the available strong lensing tools, for perhaps there is a better (and simpler) method than those we currently have in mind available for one or both of these programs.

Surveyed material

The primary source for the tools surveyed is another review of such software (Lefor, Futamase, Akhlaghi 2013). In this review, each piece of software was classified as either for research or educational purposes — here, we are interested primarily in the more rigorous treatment that the research programs will likely have, so these are the only tools we will discuss. These include: Lenstool, LensPerfect, glafic, PixeLens, SimpLens, Lensview and GRALE. (GRAVLENS has been excluded due to the fact that its dependencies can not all run on the same OS.)

The authors of the referenced review also separated the software into parametric and non-parametric, which distinguishes the model as a "forward" or "reverse" model. Forward models predict the image based on a known source and lens, while reverse models use observed images to construct mass density maps of the lensed object. We consider both here, as both methods may be useful for our purposes.

Aside from the strong lensing tools outlined in the review paper, we discuss two tools which may, in conjunction with one another, yield a valuable physics engine which gives realistic results with both numerical and graphical data:

- A black hole simulation by Github user *rantonels* (https://rantonels.github.io/starless/) which contains a general relativistic raytracer, and
- Blender, a popular open-source graphics development tool which would provide the raytracer a useful GUI as well as a python skeleton to sit the raytracer on.

Additionally, a path we've explored before and would like to look into further is a piece of software by Adam S. Bolton at University of Utah, called Bolton's Python Strong Lensing Demo. This seems to be more an educational tool than a rigorous model, but it might prove useful.

Software

We first outline the key features of the software in the table below, and discuss each piece of software and its usefulness for our purposes in more detail further down. These key features include the year the software was last updated, whether or not it is open source, whether or not it comes as an executable file, its operating system, its source language (if the source is available), and whether or not it is considered parametric or non-parametric in its modeling procedure.

Software	Year	Source?	Exec?	OS	Src. lang
Lenstool	2008	Y	N	OSX, Linux	Python*
LensPerfect	2006	Y	N	OSX, Linux	Python
glafic	2019	Y	N	(Library)	Python
PixeLens	2012	N	Y	OSX, Linux	N/A
SimpLens	2007	Y	Y	(Library)	Java
Lensview	2003	Y	Y	(Library)	Java
GRALE	2008	Y	N	OSX, Linux	C
Bolton's Lensing Demo	2012	Y	N	OSX, Linux	Python
rantonels	2017	Y	N	OSX, Linux	Python
Blender	2019	Y	Y	OSX, Linux	Python†

* Not compatible with Python 3.0
† Can be manipulated with Python scripts

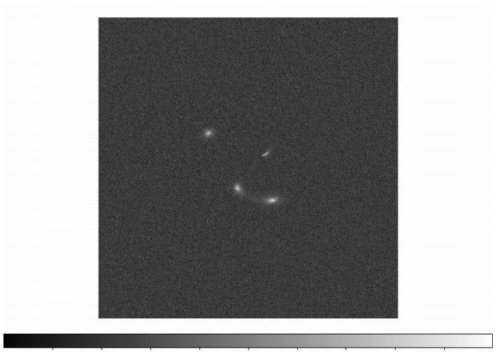
Our preference is that the software be written in Python (we're astronomers, after all), they're able to be run on OSX, and the source code is available.

In performing this review, we have found that there currently exist no programs (at least, out of those surveyed) which perform the sort of deconvolution we wish to achieve, i.e. direct image reconstruction. There exist plenty of analytical tools in this review, but their goal is to investigate strong lensing of galaxies, meaning a generating a mass map is a primary goal in reconstruction. However, we know that mass maps of exoplanets will simply yield one spherical mass and so these are not useful to us as reconstruction methods.

For this reason, we focus the discussion on those which are able to model strongly lensed systems. These programs, from the Lefor et. al. review, include only glafic and GRALE; those analytical programs which have no use currently are Lenstool, LensPerfect, PixeLens and Lensview.

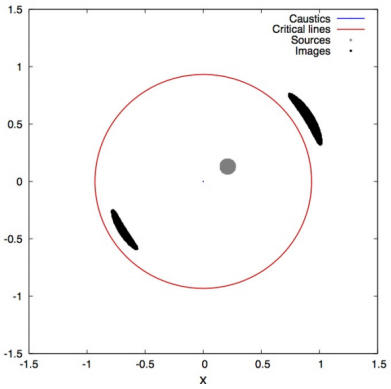
glafic

glafic takes a text file as input, filled with parameters and commands for the program, and produces a lens model with those characteristics. As can be seen to the right, glafic produces a fairly realistic image, with noise and resolution taken into account, it seems. One issue I have, however, is that there are a great many parameters that, while necessary for modeling galaxy lens systems, are more than likely irrelevant for a simple system like the SGL. This can likely be solved by simply setting those parameters (dark matter, etc.) to be negligible, but we'd like our own interface to be as simple as possible.



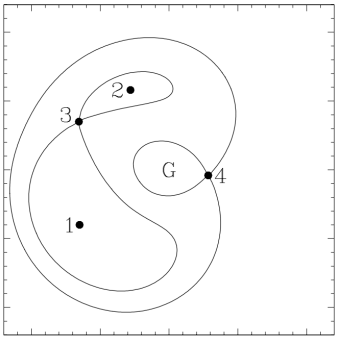
GRALE

GRALE turned out not to be what we were looking for, despite being a strong lens modeling program. An example output of GRALE is seen to the right and, as its clear to see, this is too simplistic a lens model for our purposes. It defines the features of the lensed image based on the parameters it is fed – caustics, critical points, and the lensed images themselves – but it has none of the detail we want in our own model. For this reason, GRALE is not a candidate for an SGL modeling program.



SimpLens

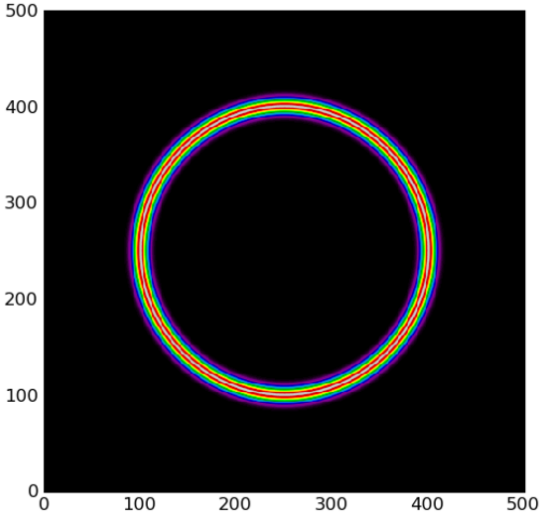
SimpLens, much like GRALE, does not produce the kind of model we aim to create. As seen to the right, it produces a qualitative plot similar to the one above, showing saddlepoint contours of a given lens. Again, we aim to produce telescope-quality images of lensed data, so models like this one are not so helpful.



Bolton's Lensing Demo

Bolton's program is an anomaly in this list, as it seems to be more of an educational tool than a simulation that would be used for this type of research. We originally tested the code when trying to create a set of quick, crude Einstein ring images for a neural net to play with, just as a proof of concept. This still may be a viable route, but there are some issues with the code itself.

The primary issue I had when trying to implement this code is that many of the commands it uses have been discarded from the Python language altogether. In order to be run properly, either these portions of the code would need to be rewritten in or ported into a newer version of Python, or we'd simply have to run the program in the older version and write any new constituent programs in the same version. This isn't a huge problem, but it's certainly a tedious solution.



Despite this, the program does seem to produce the sort of strong imaging data we're looking for, and is included in this review for that purpose. Further investigation is required.

Blender & rantonels' GR Raytracer

The combination of Blender and rantonels' GR raytracer seems the strongest candidate for a robust modeling tool for in-syzygy systems. Blender is an open-source graphics program written in Python, meaning that modules and instructions for the program can be programmed and implemented manually.

Blender uses pathtracing by default, as this is typically good enough quality for the sorts of graphical models it is often used for. This means that, on its own, Blender is not so physically accurate for gravitational lensing, as one would expect. However, if we implement a custom GR raytracing method, we may be able to change this.

This is where rantonels' code comes into play. If we implement the GR raytracer into Blender, we have a powerful tool for simulating solar in-syzygy systems. Not only do we have a graphical representation of the system, which we may be able to feed into a neural net as training data, but we may also be able to extract individual pixel data detailing spatial and spectral information from the image, based on the simulation. This is what we are ideally looking for in a model.

There is a suspicion in my mind that the program may not be built to handle the kind of data we wish to obtain from our simulations, but the fact that it is (a) open source and (b) very similar to (if not as robust as) other graphics modeling programs like Unity – the graphics engine with which Kip Thorne's "most physically accurate simulation of a black hole," as of the release date of Interstellar, was produced – makes me think otherwise.

Conclusions

This review has found that, unfortunately, a vast majority of the available strong lensing software are not fit for modeling SGL images. Most programs seek to analyze real data acquared from lensed galaxy systems, with many lenses and potentials considered at once, thus making mass maps the most relevant mode of reconstruction rather than a direct-imaging approach.

However, we do have some already established work to go off of, most notably the combination of Blender and rantonel's raytracing code, which I believe to be the most viable next step. The glafic code also seems to be promising, but parsing the parameters involved in the model will take some time.

Additionally, in reading through this review, I've found that it may have been a mistake to exclude programs that might be considered "educational," as this moniker does not necessarily distinguish them as any less of a robust simulation. In fact, Lefor et. al. notes that the distinction is arbitrary, only being made due to the fact that they haven't been cited in the literature.

The distinction, I believe, simply has to do with the sort of data produced, those being categorized as educational producing images of such systems in order to give students a visual representation of gravitational lensing. Considering we are trying to create visuals as well, the amount and type of usable data that is produced by these educational programs may be the topic of an additional useful review.